10599
Postel
[extra]

Network Working Group
Request for Comments: 357

NIC: 10599

Categories: Remote Controlled Echoing, Satellite, TELNET
References: RFC's 346, 355, 358, 318

AN ECHOING STRATEGY FOR SATELLITE LINKS

John Davidson
University of Hawaii

for

Will Crowther, BBN
John McConnell, ILLIAC
Jon Postel, UCLA

June 26, 1972

I. Introduction

As mentioned in RFC 346 ("Satellite Considerations" by Jon Postel) those interactive systems which provide echoing for full-duplex terminals over the ARPANET become more awkward to use as transmission delays increase. The reason, of course, is that a character's round trip time is added to the inherent echo delay of the server with the result that the terminal echoing appears extremely sluggish.

For a terminal separated from its server by a single satellite link, the delay will be such that even if echoing at the server were instantaneous, the latency between keying and printing of an input character will be nearly half a second. If, in addition, the character is routed thru a portion of the surface net, the delay will of course increase. It is estimated that echo delays of at least one second will not be uncommon.

This document describes a strategy which will eliminate the delay associated with simple echoing and allow the transmission delay to be hidden in the cost of computation only. This scheme is proposed as an optional addition to existing User and Server TELNETs; its use requires the explicit support of a cooperating server process.

II. Standard Echo Strategy

Echoing for locally-connected full-duplex terminals is normally provided at the server by a resident system task called the (e.g.) Terminal Handler. The Terminal Handler echoes on a one-for-one or simple replacement basis and buffers all typed input on behalf of the user process.

To let the user process operate most efficiently, the Terminal Handler should collect characters until a complete command or parameter (or whatever) has been typed. Then, presumably, the process can do some significant computing. Since the user process knows the syntax of the string it expects, it can specify to the Terminal Handler those characters which delimit completed parameters. Such characters are called wakeup characters since the user process is awakened as they are echoed.

Certain commands keyed by the user will require an output response

from the process. In order that the typed command be followed by its response and be separated from succeeding commands, the Terminal Handler must suspend echoing of user type-ahead. It can resume echoing (starting - for type-ahead - with the unechoed characters in the buffer) as soon as the process has stated (implicitly or explicitly) that it has completed the output response.

Characters which cause the Terminal Handler to suspend echoing are called break characters. They are specified by the user process based upon the syntax of the expected input. Normally break characters are also wakeup characters.

As examples:

(1) A text editor may gobble up typed English sentences every time a period or question mark is echoed. The two characters are wakeup characters only. There is no need to suspend echoing.

(2) In some systems, an ESC character is used to invoke command recognition. The user who types

COESCABCESCXYZ

should see as output

COPY (FROM FILE) ABC (TO FILE) XYZ

The ESC is both a break and a wakeup. The printout should be the same no matter how fast the user types.

The server must provide a means for each user process to communicate the following to the Terminal Handler:

(1) the set of wakeup characters,

(2) the set of break characters,

(3) which break characters should and which should not be echoed, (Some break characters - such as ESC in example 2 - should not be echoed.)

(4) completion of an output response,

(5) whether or not to echo characters. (Not echoing is useful in "hide your input" applications.)

As far as implementation, (1) and (2) could be communicated by allowing the user process to specify a 128-bit (for an ASCII device) table with 1's set for each wakeup character, and another table with 1's set for each break character. This approach becomes fairly expensive in terms of core memory as the number of terminals becomes large; the system must store these bit tables itself since in most cases the user process will not be in core while echoing is being done by the Terminal Handler.

To reduce the storage requirements, the system can make known to all its programmers a limited number, say 4, of supported break character sets, or classes (and 4 - possibly the same - wakeup character sets). Thus the programmer can choose (e.g.) break characters for his process from, for example

(a) alphanumeric characters,

(b) punctuation characters,

(c) echoable control characters (including the bell and CR, etc.), or

(d) non-echoable control characters (Control-C, etc.),

by specifying in a system call which break set(s) should be used. This requires no more than 4 bits of system storage per terminal, and a single table to identify the set(s) to which each of the 128 possible ASCII characters belongs.

For the user process to communicate (3) to the Terminal Handler (which break characters should and which should not be echoed), the process can specify another 4 bit field with 1's set for those break classes whose members should be echoed. For the 4 classes above, only 3 bits would be required since members of class (d) are defined to be non-echoable.

To communicate the completion of an output response (4), the user process could issue an explicit system call; or, the Terminal Handler could assume completion when the user process requests input of the first character following the break.

"Hide your input" (5) would be communicated by a system call which specifies either

(a) "break on every character and don't echo any break characters", or, for example

(b) "don't echo anything and break on punctuation, or any control character" for an alphanumeric password,

depending on the syntax of the expressions to be hidden.

III. Definitions

Several new terms need to be defined, some of which are direct extensions of the terms used in the "standard echo strategy" description. There is no reason to insist that the four buffers presented all be implemented as individual constructs; they are logically separated for clarity in the discussions which follow.

Remote Controlled Echoing (RCE)

This is the name for the echo strategy described in this document. Echoing will be controlled by the (remote) server but performed by the User TELNET.

Input Buffer

This is a logical buffer used by a User TELNET to hold characters in sequence as they are received from the terminal keyboard (after they have been converted to NVT characters).

Transmission Buffer

This is a logical buffer used by a User TELNET to hold NVT characters which have been typed but have not yet been transmitted to the server.

Output Buffer

This is a logical buffer used by a User TELNET to hold the NVT characters received from the server.

Print Buffer

This is a logical buffer residing in the User TELNET from which characters will be sent in sequence to the terminal printer. (The output buffer contains NVT characters which may have to be converted to characters employed by the actual terminal.)

Break Classes

The 128 possible (7-bit) ASCII characters employed by the Network Virtual Terminal can be partitioned into several quasi-equivalence classes (for example alphabetic, numeric, punctuation characters, etc.). These classes can be defined so that each character is a member of at least one class, although it may belong to more than one.

A server process may indicate to a User TELNET that certain of these classes (or all, or none) are to be considered break classes. That is, a break class is an equivalence class which is of special significance to the server process. In terms of the discussion of section II, the server recognized 4 equivalence classes any combination of which might be designated as break classes by a particular process.

The RCE implementation will have more than 4 equivalence classes (perhaps as many as 8) to provide more flexibility in the choice of break character sets.

Break Character

A break character is any member of any break class. In the standard echo strategy, receipt of a break character meant "stop echoing"; with RCE, the break character means "stop moving characters from the input buffer to the print buffer".

A break character has no relation to the NVT break control code.

Break Action

Two break actions are possible:

(1) a break character encountered in the input buffer is moved to the print buffer at the appropriate time, or

(2) a break character encountered in the input buffer is not moved to the print buffer.

The server process will specify which break action should be followed. (The two actions correspond to echoing or not echoing the break character.)

IV. Description

(This description is written in terms of the TIP which, of course, embodies a User TELNET.)

Remote Controlled Echoing is an attempt to remove the echo responsibility from the Terminal Handler and to push it off into the TIP; wakeup processing is still handled at the server. The process' interface (system calls, etc.) to the server's Terminal Handler need not change, but the (abbreviated) Terminal Handler (actually a Server TELNET) must find a way to relay the process' echo requirements to the TIP. It does this with TELNET commands and control information. System calls and echo parameters (break classes, etc.) peculiar to a particular serving Host must be interpreted by the Server TELNET at that Host and reissued to the TIP in terms of these new TELNET commands.

Character Flow

Refer to figure 1. A character received from a full-duplex terminal will be converted to its NVT equivalent and placed in both the transmission and input buffers. The TIP's transmission strategy determines when it will be removed from the transmission buffer; the server's RCE control commands dictate when it will be removed from the input buffer.

A character received from the server will be placed in the output buffer.

Of the three labeled paths DISCARD, ECHO and OUTPUT, exactly one is enabled at all times. RCE commands dictate which one. Thus characters may

(DISCARD:) be removed in sequence from the input buffer and discarded,

(ECHO:) be removed in sequence from the input buffer and placed in the print buffer, or

(OUTPUT:) be removed in sequence from the output buffer and placed in the print buffer.

From the print buffer they will be converted from NVT characters and be immediately sent to the terminal's printer.

Commands: Server to User

The following are the proposed TELNET commands sent by the server process to the TIP. Commands (2) thru (5) should not be sent if RCE is not being used.

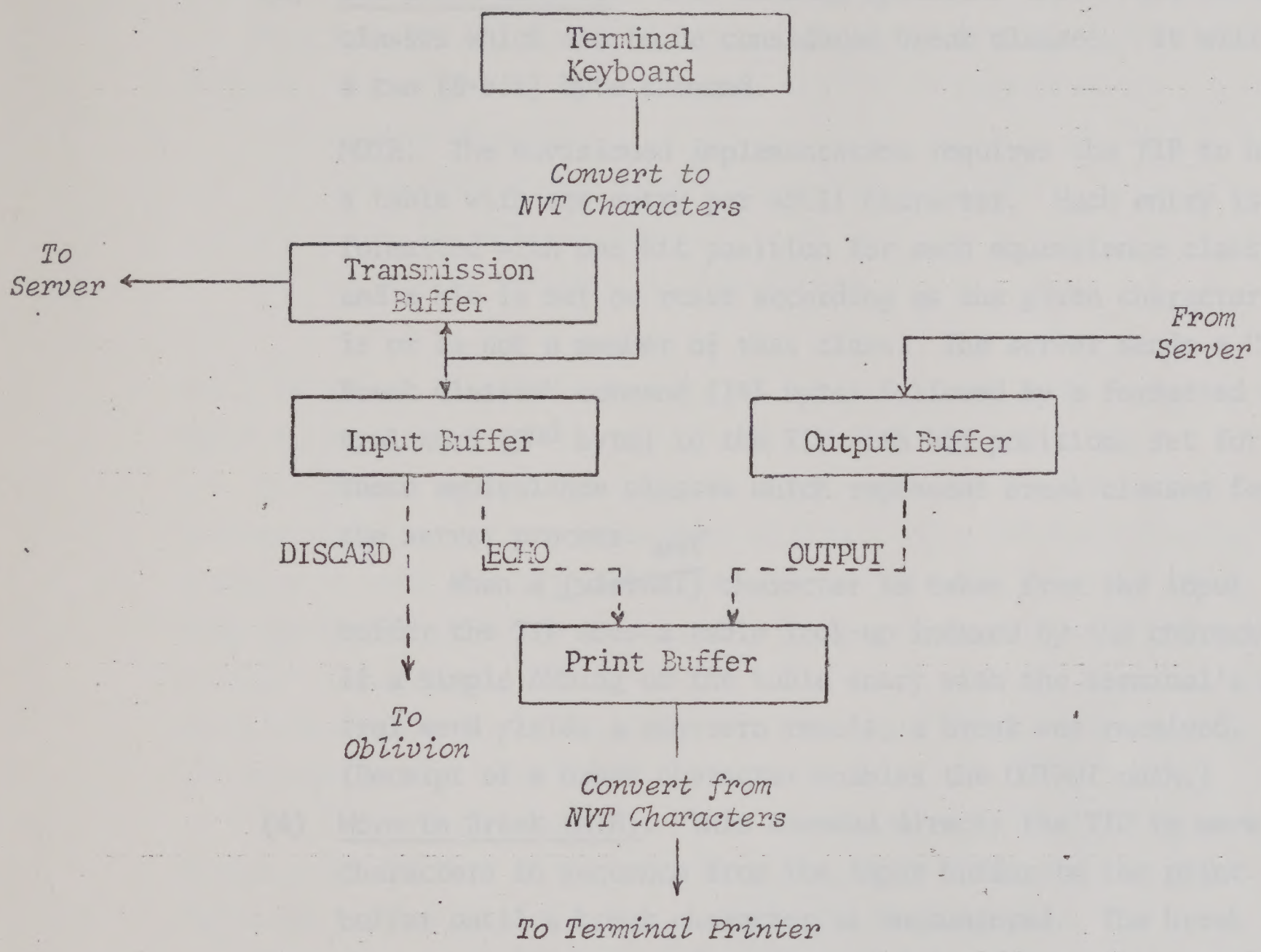

Figure 1. Character Flow within the TIP

(1) Use Remote Controlled Echoing. The server asks the TIP to employ the echo strategy described in this document. The TIP can respond either YES*(I will use it) or NO.

(2) Set Break Action. This is actually 2 commands. The server can set the break action to echo or not echo a break character.

(3) Set Break Classes. This command specifies those equivalence classes which are to be considered break classes. It will be a two (8-bit) byte command.

NOTE: The envisioned implementation requires the TIP to have a table with one entry per ASCII character. Each entry is formatted with one bit position for each equivalence class, and a bit is set or reset according as the given character is or is not a member of that class. The server sends a "Set Break Classes" command (1st byte) followed by a formatted control word (2nd byte) to the TIP with bit positions set for those equivalence classes which represent break classes for the server process. NVT

When a ~~(virtual)~~ character is taken from the input buffer the TIP does a table look-up indexed by the character. If a simple ANDing of the table entry with the terminal's control word yields a non-zero result, a break was received. (Receipt of a break character enables the OUTPUT path.)

(4) Move to Break (MTB). This command directs the TIP to move characters in sequence from the input buffer to the print buffer until a break character is encountered. The break character will be moved or discarded depending on the previously-specified break action. (Essentially, MTB enables the ECHO path.)

(5) Delete to Break (DTB). This command directs the TIP to remove characters in sequence from the input buffer and discard them until a break character is encountered. The break character will also be discarded. This provides a convenient mechanism for hiding a user's input. (DTB enables the DISCARD path.)

---

*It is suggested that the response YES also be "Use RCE" to eliminate race conditions.

Commands: User to Server

The user may send (via the TIP) a request to the server process requesting that it "Use Remote Controlled Echoing" to which the server must respond "YES" or "NO".

General Operation

Very simply, the Remote Controlled Echoing strategy works as follows: The Server TELNET will tell the TIP to (essentially)

(1) echo a message,
(2) print the process' response to that message,
(3) echo the next message,
(4) print the response to that message
. . .

etc., to effect an orderly listing of inputs and responses much as would be imposed when using a half-duplex device.

The actual interaction depends on the control commands. When a terminal-serving process is invoked on behalf of a TIP user, the Server TELNET will send the "Use RCE" command; the TIP will respond "YES". Then the Server TELNET will send the "Set Break Action" and "Set Break Classes" commands to properly reflect the break strategy requested by the terminal-serving process. Lastly the Server TELNET will send an MTB command. This enables the ECHO path.

The TIP removes characters from the input buffer and places them in the print buffer. When it encounters a break character, it performs the break action specified, and enables the OUTPUT path.

Characters are then moved in sequence from the output buffer to the print buffer. When an MTB (or, DTB) is encountered, it is discarded and the ECHO (DISCARD) path is enabled.

Etcetera.

The Server TELNET may change the break action or break classes after any interaction, but should normally do so prior to sending the MTB or DTB commands. It should only send an MTB (DTB) after all process output from the previous message has been sent.

Why Does This Work?

The RCE strategy described above produces the correct result at the user's terminal because it is in essence the same scheme used by the Terminal Handler which normally provides the echoing at the serving site. Initially, characters are echoed as they are typed; when a break character is keyed, echoing is suspended. If the process produces any output response, it is printed before echoing of subsequent input.

Echoing of the next command begins, if there is type-ahead, with the characters in the input buffer, and even if the input buffer is emptied, immediate echoing of keyed input from the terminal is provided since the ECHO path remains enabled up to a break.

An Example

(In the following, we assume all break characters are also wakeup characters and (carelessly) treat the two interchangeably.)

Suppose a TIP user attempts to login to a remote server with the properly formatted message

LOG NAME PASSWORDⒸⓇ

and that the Server TELNET has requested the use of RCE. Presuming that the break (and wakeup) character sets are appropriately defined to include space and CR (and that the break action specifies they should be echoed), the primary sequence of RCE control commands which will drive the TIP to produce the correct printout at the user's terminal is

(1) MTB (to print "LOG "), and since the space is a break character,
(2) MTB (to print "NAME "),
(3) DTB (to delete "PASSWORDⒸⓇ"*), and perhaps a message followed by
(4) MTB (to reenable the echo path).

We investigate in some detail how interaction at the process/Server TELNET interface causes these commands to be sent to the TIP.

When the EXEC is invoked, it issues a system call to set its break classes. The Server TELNET interprets the system call in terms of the classes supported by RCE, and sends the appropriate two-byte "Set Break

---

*See Section VI number 11

Classes" (SBC) command to the TIP. A space is among the characters of the break set.

The EXEC asks for input, so the Server TELNET sends MTB ((1) above). We presume the EXEC blocks until some input is available.

The EXEC is awakened when the first space arrives; it recognizes the LOG command to be a call upon the LOGIN subsystem which it (promptly!) invokes.

The LOGIN process issues a system call to set its break classes (this time both space and CR are included, and, as before, the Server TELNET forwards the command as an SBC). Then it asks for input (so the Server TELNET sends MTB ((2) above)), and blocks until the second space arrives.

When the LOGIN process has verified the existence of a user with name NAME, it issues a system call to suppress printing of the next parameter (the password). In compliance, the Server TELNET sends DTB ((3) above).

Once the password has been examined and verified, a message like

CRLF LOGIN COMPLETEDCRLF

can be sent, followed by a request for input. The Server TELNET thus forwards an MTB ((4) above) and the sequence is completed.

Another Example

Suppose in the above example the user had typed

LOG NAMECR

When the LOGIN process regained control, it would have noted that the break was a CR instead of a space. It then could have issued

LFpassword =

which the Server TELNET would follow (when LOGIN requests print suppression) with DTB. When the TIP had finished its output, the DISCARD path would be enabled and the user's terminal would have printed:

LOG NAME
password =

with the cursor positioned as shown. The TIP will hide the characters of the password.

Another Example

Suppose a user were using a text editor, TEXT, to create a source file of English sentences. The TEXT subprocess might allow only non-formatting control characters (e.g., "Control-C") as break characters. The RCE strategy would allow the user to receive immediate echoing for all his input until he keyed such a control character.

V. Discussion

The Remote Controlled Echoing Strategy is designed to provide echoing for a full-duplex terminal as if it were locally connected to its server. The effect of the long transmission delays will only be evident as an increase in the processing performed at a break. Only in the most interactive systems will such delay be consistently noticeable. For example if a user invokes a long FORTRAN compilation, the fact that its start is delayed for half a second will not normally be evident.

Furthermore, users who are able to type several messages ahead may only notice a processing delay as a result of the first break-interaction; both transmission and processing of successive messages may occur during the printing of "echoes" and responses to previous messages.

Transmission Considerations

In the standard echoing scheme, characters are buffered at the server as they are keyed. But the user process does not see them until a wakeup character has been typed. This means a TIP using RCE could buffer characters in the transmission buffer until a wakeup occurs and then send off the whole bunch. Unfortunately we have chosen, for simplicity, to keep all knowledge of wakeup characters at the serving site. This means that the TIP may buffer beyond a wakeup [if it is not also a break] and delay the process from doing some useful work. However, since in this case no output is expected from the process, no

noticeable delay is visible to the user, except that the next break interaction may take a little longer.

If the TIP chooses to buffer input before transmission, it will transmit at least at every break character. The server should be able to instruct the TIP to transmit more often if it is appropriate.

An Example

Conversational output linking is an example where transmission strategy is separate from the break and wakeup strategies. Transmission should occur on every character so that the character can be promptly printed at each linked terminal, but no break or wakeup need occur until a special escape character is typed (this reawakens the EXEC, for example).

Conversational output linking also introduces another funny:

Unsolicited Output

What happens when the ECHO(or DISCARD) path is enabled, but the input buffer is empty (i.e. immediate echoing is occurring) and something arrives in the output buffer? This "something" cannot be a response to a previously keyed command, or the ECHO path would be disabled. (This proof is left to the reader!) It is most likely a system message like

CR LF SYSTEM WILL GO DOWN IN 5 MINUTES CR LF

or, a character arriving from another linked terminal.

Since such output does not fit neatly into our RCE scheme, the following kludge is proposed. Unsolicited output should cause the OUTPUT path to be enabled. The occurrence of either an MTB (DTB) or empty output buffer will reenable the ECHO(DISCARD) path.

VI. Orthogonal Issues

Several other mechanisms may reasonably be incorporated within this proposed addition to TELNET. The problems which need some further discussion include:

1) Some means should be provided for the server to clear the input, transmission, print, and output buffers.

2) Some means should be provided for the user to immediately clear the output buffer (i.e. suppress printing of lengthy output).
3) The server may want to ask about the number of characters remaining to be printed.
4) A special tag character may be required to note where clearing of the input buffer occurred.
5) The TIP's transmission strategy should be specifiable by the server; perhaps a "Set Wakeup Classes" command should be implemented.
6) The server should be able to cause the TIP to break on the n^{th} input character regardless of whether a break character has been encountered.
7) Should the TIP or the server be responsible for properly echoing a formatting control character such as a TAB?
8) The proper equivalence classes of ASCII characters have to be finalized.
9) How should a CR be echoed?
10) Should one-for-one echo replacement (e.g. "$" for ESC) or multi-character substitution (e.g. "↑A" for Control-A) be provided by the TIP?
11) The proposed DTB command directs the TIP to also discard the delimiting break character. Should the break character discard rather be dependent on setting the Break Action to "don't echo" before sending the DTB?

Several of these issues will be addressed in RFC 358.

VII. Conclusion

This document has presented a proposed optional addition to the User TELNET. The next step is to perform some simulations and to incorporate RCE into an experimental version of TELNET.

No recommendation is made that the current TELNET be discarded. For example RCE should not be used for those half-duplex devices which perform their own "echoing". If RCE is adopted as an alternate means of echoing, changes to those TELNETs choosing not to implement it should

be minimal. Switching from RCE to the current echo mechanism is an immediate result of either user or server sending a "You Echo" (or "I'll Echo").

Much of the machinery required to implement RCE already exists at the interface between the server process and its Terminal Handler or Server TELNET. This means that no server processes need be changed, but that the process' means of specifying break classes, break actions and echoing conventions must be interpreted by the Terminal Handler and reissued to the TIP in terms of the corresponding RCE commands.